



www.thalesgroup.com

xHTML Documentation Generation

 **Kitalpha**



OPEN
Version 1.0.0

THALES



- 1 Introduction**
- 2 User Perspective**
- 3 Developer Perspective**

Context

- A model contains information that can be consulted outside the modeling context (e.g., for share of information)

Need

- Generating documentation from a model in a standard form

Objective

- Producing xhtml documentation from a model
- Providing a xhtml documentation framework customizable for a model or modeling workbench
- Providing developers with tools to extend and customize a documentation generation

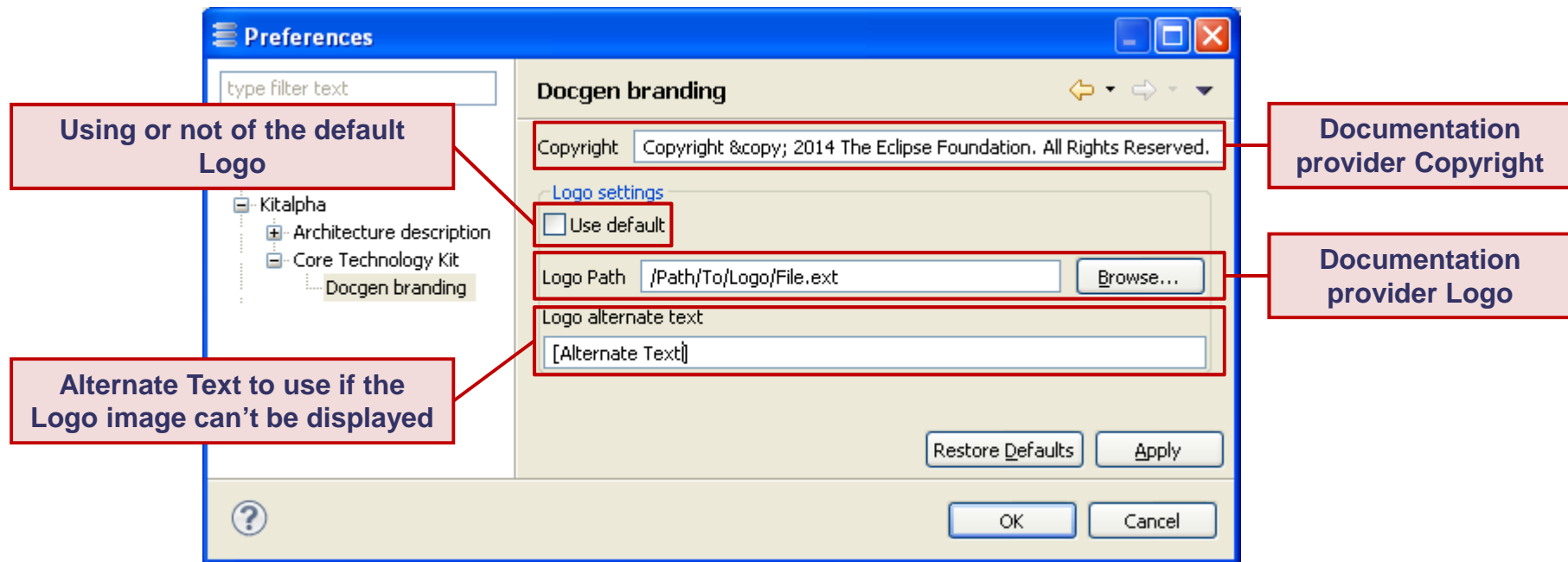
This document is not to be reproduced, modified, adapted, published in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.



- 1 Introduction
- 2 User Perspective
- 3 Developer Perspective

Branding

- Customizing the documentation provider copyright and logo



Objective

Allowing documentation providers to use their own copyright and logo

Actions

Modifying branding data in the Docgen preferences

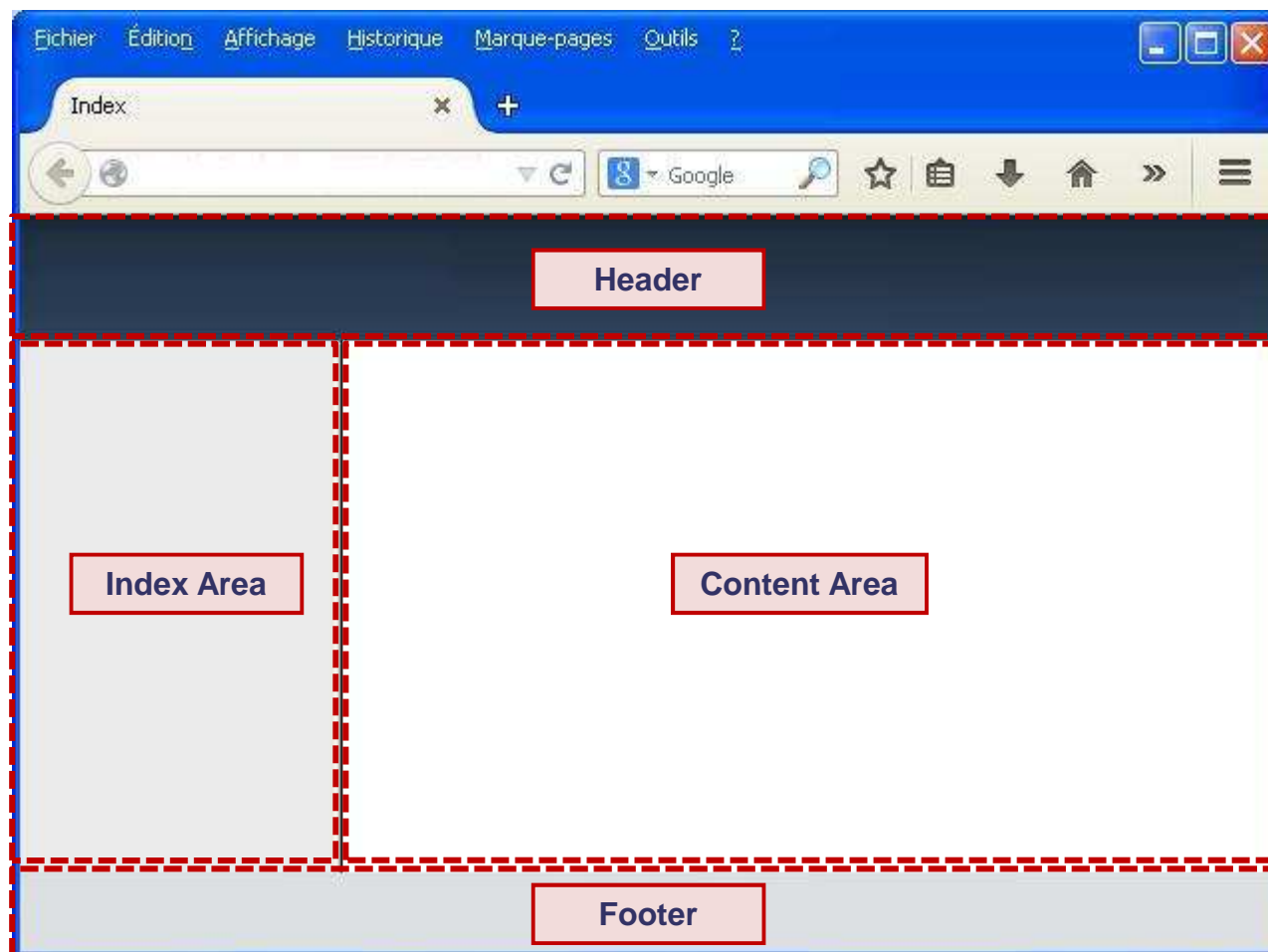
Branding

- Customizing the documentation provider copyright and logo

Structure

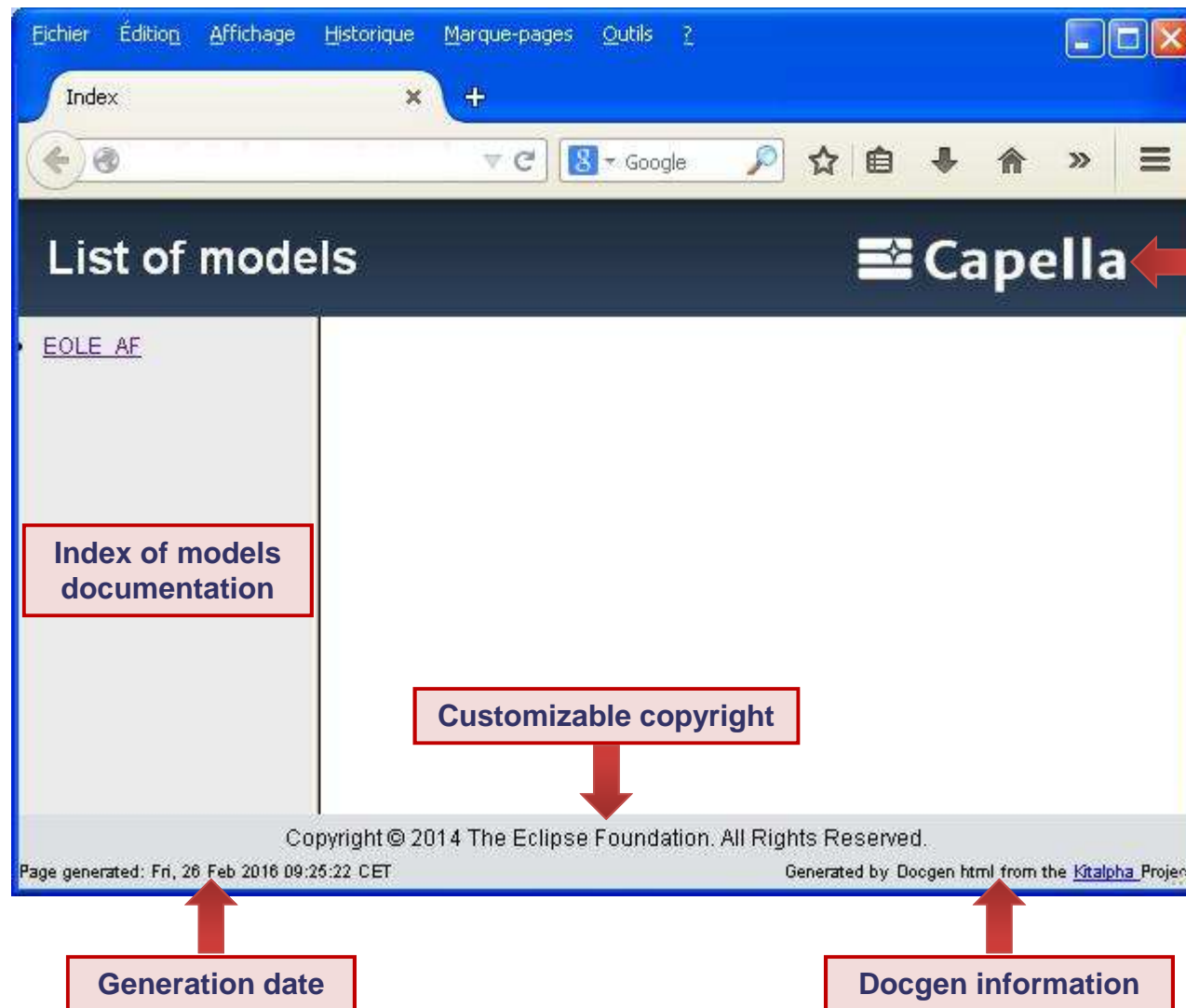
- Sharing the same tools, structure and style over all pages
- Cross-Navigating on several models documentation

I. Generic template



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

II. Several models documentation



III. One model documentation (1/2)

The screenshot shows the Capella documentation interface for the EOLE_AF model. The interface is divided into several sections:

- Header:** Displays the model name "EOLE_AF" and the Capella logo.
- Left Panel (Documentation index):** A tree view showing the model's structure, including "EOLE_AF", "Operational Analysis", "System Analysis", "System Functions", "Data", "System Context", "EOLE", "Actors", "Missions", "Logical Architecture", "Physical Architecture", and "EPBS Architecture".
- Right Panel (One model element documentation):** Displays the documentation for the "Missions" element. It includes a "Content" section with the text "Provide meteo data" and an "Owned diagrams" section showing a diagram titled "[MB] EOLE - Missions Blank".

The diagram in the "Owned diagrams" section shows a sequence of actors and a process:

```
graph LR; ScientificUser[Scientific user] --> ProvideMeteoData((Provide meteo data)); ProvideMeteoData --> Atmosphere[Atmosphere];
```

Page generated: Fri, 26 Feb 2016 14:20:09 CET. Copyright © 2014 The Eclipse Foundation. All Rights Reserved. Generated by Docgen html from the Kitalpha Project.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

IV. One model documentation (2/2)

The screenshot shows the Capella software interface with the following elements:

- Menu Bar:** Fichier, Édition, Affichage, Historique, Marque-pages, Outils, ?
- Search Bar:** Google
- Capella Logo:** Capella
- Left Sidebar (Tree View):**
 - EOLE_AF
 - EOLE_AF
 - Operational Analysis
 - System Analysis
 - System Functions
 - Capabilities
 - Interfaces
 - Data
 - System Context
 - EOLE
 - Actors
 - Missions
 - Logical Architecture
 - Physical Architecture
 - EPBS Architecture

- Main Content Area:**
- MissionPkg**
 - EOLE_AF > EOLE_AF > System Analysis > Missions
- No description.
- Content**
 - Provide meteo data
- Owned diagrams**
 - [MB] EOLE - Missions Blank
- Diagram showing a use case 'Provide meteo data' (M) connected to 'Scientific user' and 'Atmosphere'.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

Branding

- Customizing the documentation provider copyright and logo

Structure

- Sharing the same tools, structure and style over all pages
- Cross-Navigating on several models documentation

Search

- Searching concepts and references to them in a model documentation

The screenshot shows the Capella documentation search interface. The browser window has a menu bar with 'Fichier', 'Édition', 'Affichage', 'Historique', 'Marque-pages', and 'Outils'. The address bar shows 'Index'. The main content area displays the 'EOLE_AF' model page. The left sidebar contains a list of indexed concepts, including 'Acquire data', 'Acquire data CapabilityPkg', 'Acquire data FC', 'Acquire images', 'Acquire images FC', 'Atmosphere', 'Balloon', 'Batch Processing', 'Board Computer', 'Boolean', 'Broadcaster', 'Byte', and 'Camera'. The right sidebar shows the 'Content' section with a list of topics: 'Operational Analysis', 'System Analysis', 'Logical Architecture', 'Physical Architecture', and 'EPBS Architecture'. The footer contains copyright information and page generation details.

Go back to model documentation

Filter text field

Back to Model

Search index - Indexed Concepts

EOLE_AF
SystemEngineering
EOLE_AF > EOLE_AF

No description.

Content

- Operational Analysis
- System Analysis
- Logical Architecture
- Physical Architecture
- EPBS Architecture

Copyright © 2014 The Eclipse Foundation. All Rights Reserved.
Page generated: Fri, 26 Feb 2016 14:20:09 CET
Generated by Doogen html from the [Kitalpha](#) Project

The screenshot displays the Capella search interface. The search criteria 'Bal' is entered in the search bar. The search results list includes 'Balloon', 'Global Capability Realization', 'Global Data Capability Realization', and '[PrimerItemCI] Balloon'. The search results for 'Balloon' are shown, including references to one concept and a list of concepts referenced in the title and list.

Search criteria

Search criteria: Bal

Search result - Concepts list

- Balloon
- Global Capability Realization
- Global Data Capability Realization
- [PrimerItemCI] Balloon

Search result - References to one concept

Balloon

Concept referenced in title

- Capella - EOLE_AF - ConfigurationItem [PrimerItemCI] Balloon
- Capella - EOLE_AF - Part Balloon
- Capella - EOLE_AF - PhysicalComponent Balloon
- Capella - EOLE_AF - Part Balloon

Concept referenced in list

- Capella - EOLE_AF - ConfigurationItem [PrimerItemCI] Balloon
- Capella - EOLE_AF - Part Balloon

Copyright © 2014 The Eclipse Foundation. All Rights Reserved.
Page generated: Fri, 26 Feb 2016 14:20:09 CET
Generated by Doogen html from the [Kitalpha](#) Project

Branding

- Customizing the documentation provider copyright and logo

Structure

- Sharing the same tools, structure and style over all pages
- Cross-Navigating on several models documentation

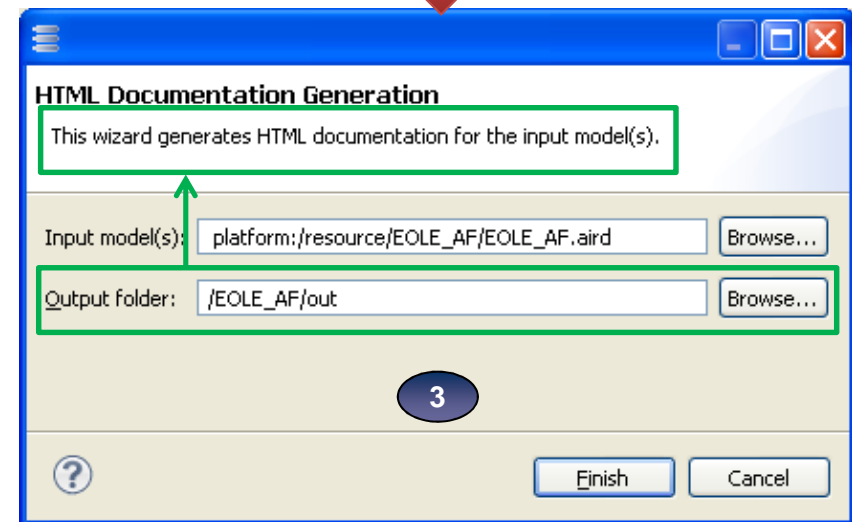
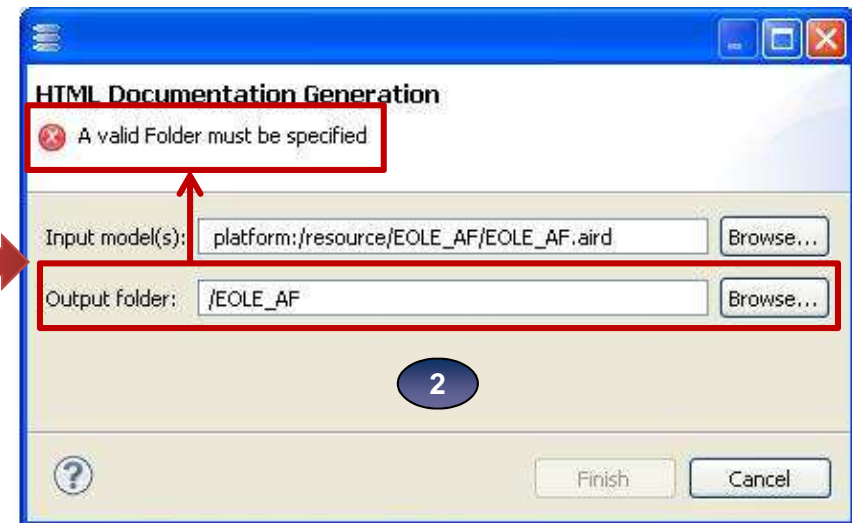
Search

- Searching concepts and references to them in a model documentation

Generation

- Documentation generation by wizard or in headless mode
- Full or partial generation

I. Default information



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

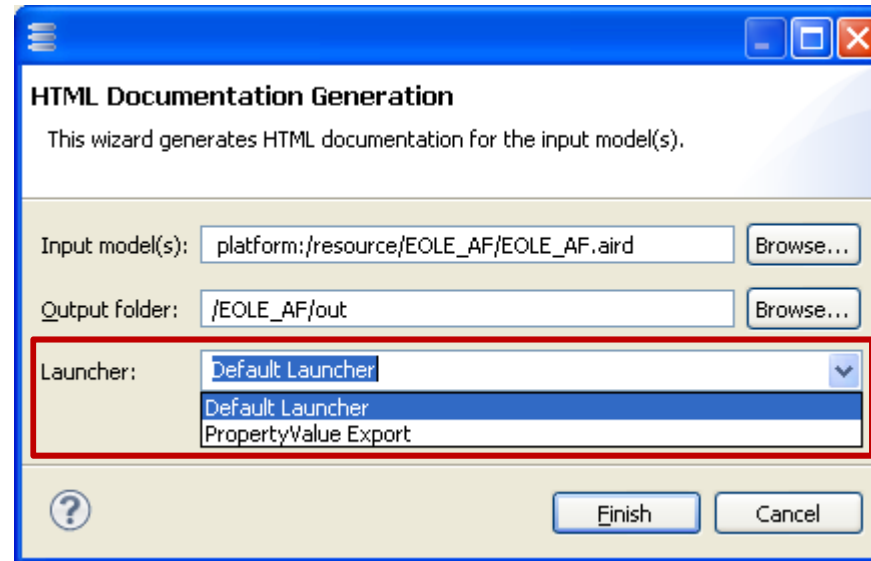
Objective

- Providing integrated documentation generation
- *Ex. Docgen for Capella*

Actions

- Displaying the generation wizard
- Choosing the output folder
- Launching the documentation generation

II. Extensions Launchers



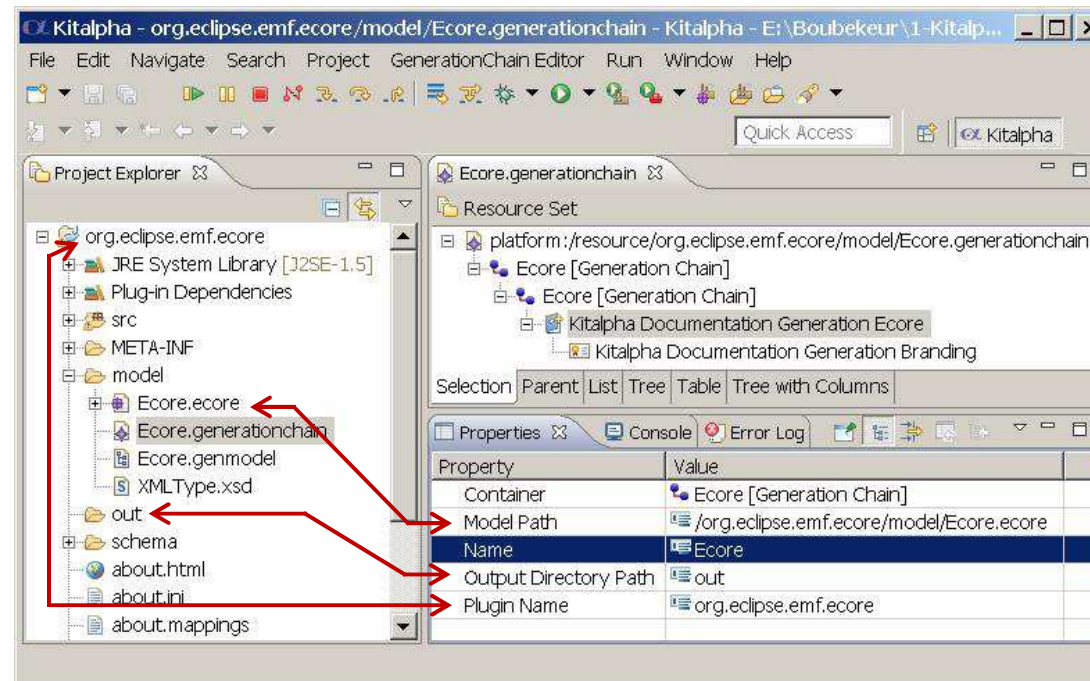
Objective

Generating extended / Customized model documentation

Actions

Choosing one of the available launchers

- "Default Launcher" is the default generation launcher
- " PropertyValue Export" is an example of extended generation



Objective

- Providing headless documentation generation
- *Ex. Docgen for Ecore*

Actions

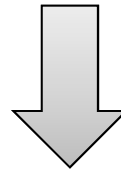
- Development time
 - Developing an extension of the generation chain
- Using time
 - Creating a Generation chain
 - Adding a configuring documentation generation chain element
 - Running the generation chain (manually or programmatically)



- 1 Introduction
- 2 User Perspective
- 3 Developer Perspective

Foundations

A framework for xhtml documentation generation



Customization for specific context

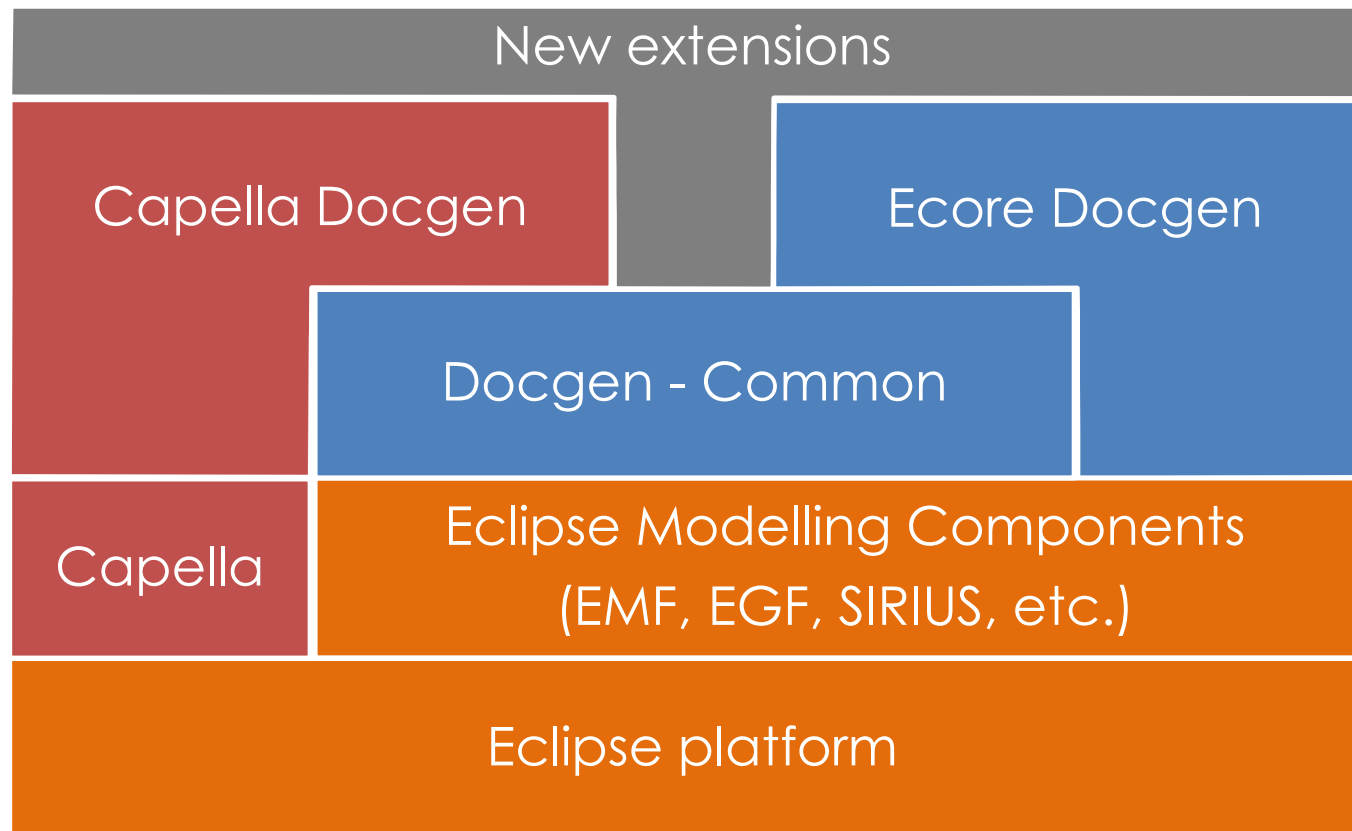
Available



For Ecore models



For Capella models



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

Implementation

Plugin name	org.polarsys.kitalpha.doc.gen.business.core, org.polarsys.kitalpha.doc.gen.business.core.ui
Java Package	All
Java class	All

Extensions point

Name	Plugin	Schema
Concepts Helper	org.polarsys.kitalpha.doc.gen.business.core	conceptshelper.exsd
Description	This extension point allows to decide if a model element has to be handled by the generation of search index or not. Extension must contribute with a helper that is an implementation of the Java interface IConceptHelper.	

EGF Factory components

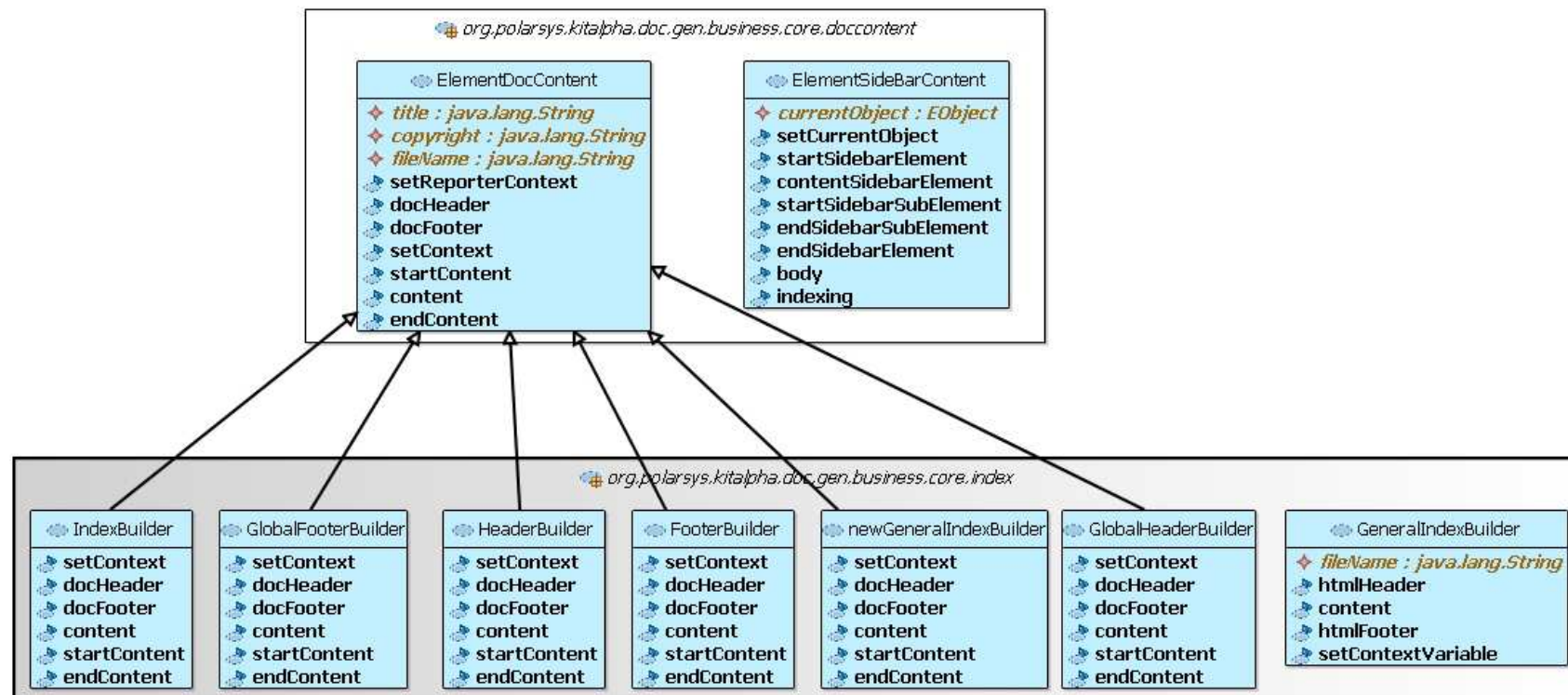
Name	Description
HTMLDocGenCommon.fcore	Contains all EGF pattern and factory component facilitating the development of an xHTML generation for a given model.

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns



1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns

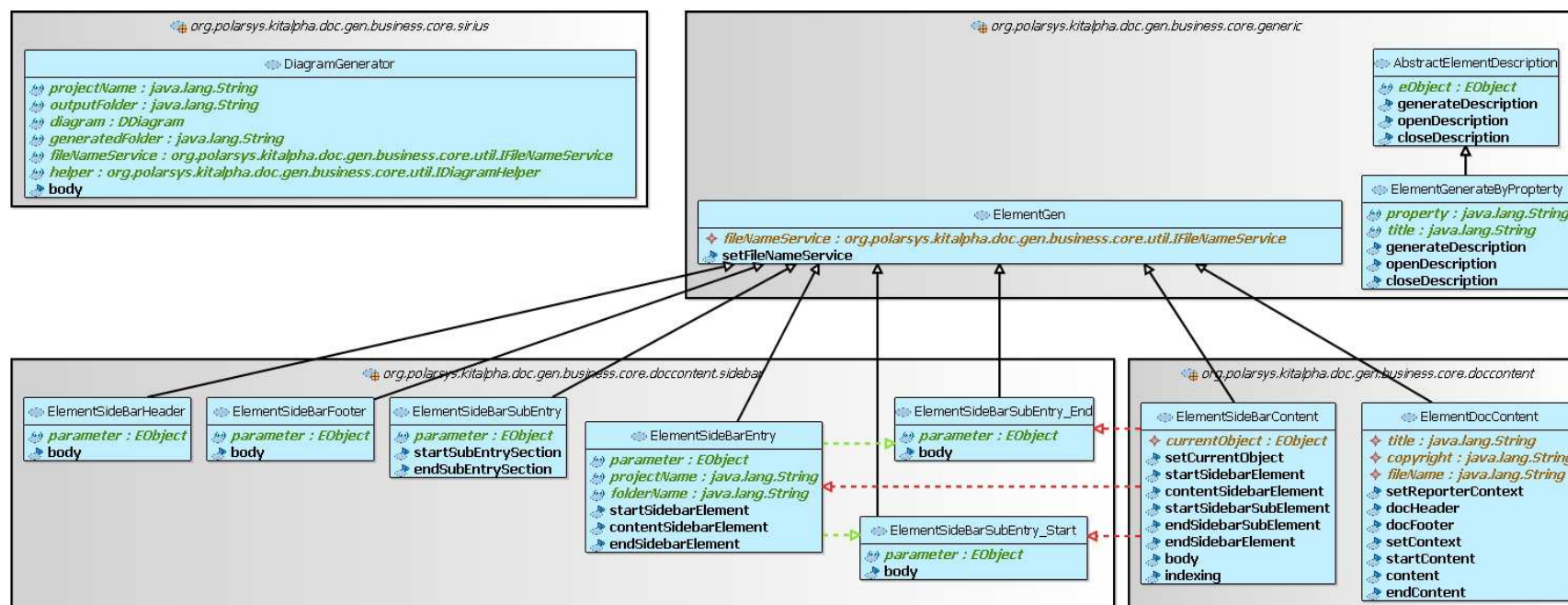
Name	org.polarsys.kitalpha.doc.gen.business.core.index			
Description	Contains patterns to generate index html files. Note that there are three kind of index files: model documentation index file; documentation general index file and models index file.			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
IndexBuilder	JET	--	ElementDocContent	Generate the index.html. This file is the index of one model documentation
GeneralIndexBuilder	JET	--	ElementDocContent	Generate the modelindex.html file. This file contains links to model index.html.
HeaderBuilder	JET	--	ElementDocContent	Generate the header.html. This file is the header of one model documentation
FooterBuilder	JET	--	ElementDocContent	Generate the footer.html. This file is the footer of one model documentation
newGeneralIndexBuilder	JET	--	ElementDocContent	Generate the general index.html. This file is the index generated in the output folder, it is independent from models html files.
GlobalHeaderBuilder	JET	--	ElementDocContent	Generate the header.html. This file is the general header used by the general index.html file.
GlobalFooterBuilder	JET	--	ElementDocContent	Generate the footer.html. This file is the general footer used by the general index.html file.

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns (1/4)

Name	org.polarsys.kitalpha.doc.gen.business.core.generic			
Description	This pattern library contains some abstract and reused patterns			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
ElementGen	JET			A hight level pattern containing a variable that will host the file name service to use. A File Name Service is a Java class implementing the naming convention to use in a given generation.
AbstractElementDescription	JET	EObject		Contains tree methods that can be used to generate one element description. There is no implementation in these methods.
ElementGenerateByPropterty	JET	String (Property, title)	AbstractElementDescription	Generates a section of an html page in a <p> </p> block

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns (2/4)

Name	org.polarsys.kitalpha.doc.gen.business.core.doccontent			
Description	---			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
ElementDocContent	JET		ElementGen	An abstract pattern containing all method allowing the generation of an html page. This patterns contains the methods generating the header and the footer of the html pages and initialize some information mandatory to the generation like the file name that the reporter will use to serialize the html file on the hard drive.
ElementSideBarContent	JET		ElementGen	This patterns does the same as the ElementDocContent but for generating the sidebar entries.

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns (3/4)

Name	org.polarsys.kitalpha.doc.gen.business.core.sirius			
Description	Contains the pattern used to export Sirius diagram to JPG pictures. These pictures are used in the generated html page.			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
DiagramGenerator	JET	String (projectName, outputfolder, generatedFolder) DDiagram (diagram) IFilenameService (fileNameService) IDiagramHelper (helper)		This pattern handle Sirius diagrams. It allows to: 1- Export Sirius diagrams to images files. 2- Calculate clickable areas on the exported images.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

1 Index generation

2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns (4/4)

Name	org.polarsys.kitalpha.doc.gen.business.core.doccontent.sidebar			
Description	Contains all pattern necessary to generate a Sidebar			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
ElementSideBarHeader	JET	EObject	ElementGen	Generate sidebar header. It contains links to go back to models index and search index. It contains too expand all / collapse all tools.
ElementSideBarFooter	JET	EObject	ElementGen	Generate the end of the sidebar.html file. This section contains a Javascript script that declare an element as a Treeview.
ElementSideBarEntry	JET	Eobject (parameter) String (projectName, folderName)	ElementGen	Generate one side bar element entry. It is an element,
ElementSideBarSubEntry	JET	EObject	ElementGen	This pattern is used when a sidebar element has sub entries. So this pattern generate an wherein the sub entries will be generated,
ElementSideBarSubEntry_Start	JET	EObject	ElementGen	Generate an tag
ElementSideBarSubEntry_End	JET	EObject	ElementGen	Generate an tag

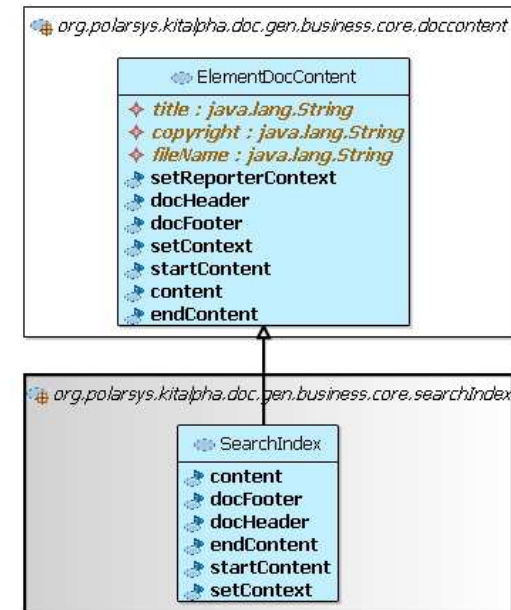
1 Index generation

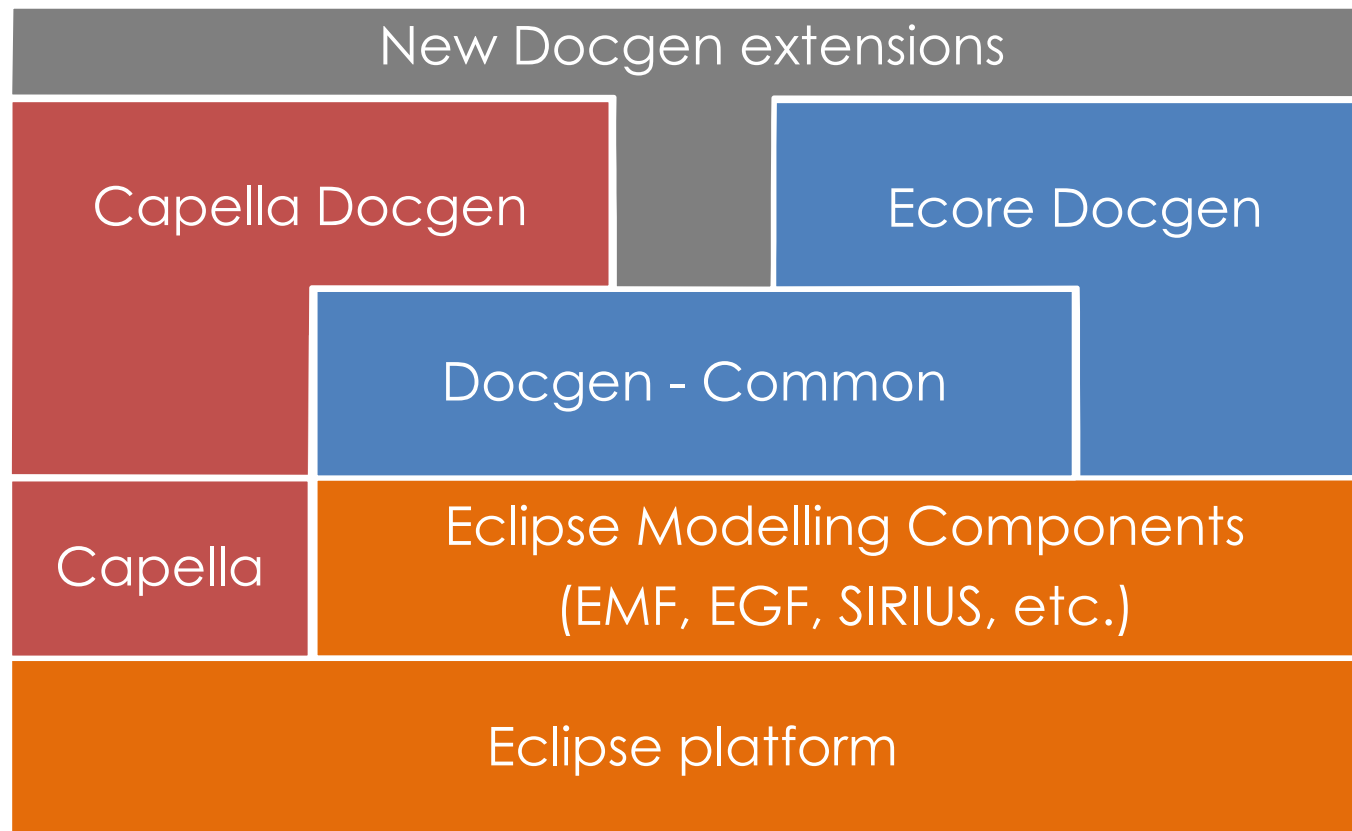
2 Content generation

3 Search index generation

EGF Pattern Libraries and Patterns

Name	org.polarsys.kitalpha.doc.gen.business.core.searchIndex			
Description	Contains the pattern generating the search index html file			
Patterns				
Name	Nature	Parameter	Super Pattern	Description
SearchIndex	JET		ElementDocContent	Generate the searchindex.html page.





This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

Ecore Documentation Generation


Cost

[Search Index](#) | [Back to Index](#)

- Cost
 - Cost
 - Justification
 - InfoLog
 - Info

Package : Cost

Cost



Attributes

	Name	Type	Changeable	Required	Derived	Cardinality	Description
	id	EString	true	true	false	1..1	the unique identifier for this element [source: Capella study]
	sid	EString	true	false	false	0..1	the unique system identifier for this element
	name	EString	true	false	false	0..1	The name of the NamedElement [source: UML superstructure v2.2]
	visibleInDoc	EBoolean	true	false	false	0..1	none
	visibleInLM	EBoolean	true	false	false	0..1	none
	summary	EString	true	false	false	0..1	Summary of the element [Capella study]
	description	EString	true	false	false	0..1	Description of the Capella element [Capella study]
	review	EString	true	false	false	0..1	Review description on the Capella element
	objective	EFloat	true	false	false	0..1	
	max	EFloat	true	false	false	0..1	
	value	EFloat	true	false	false	0..1	

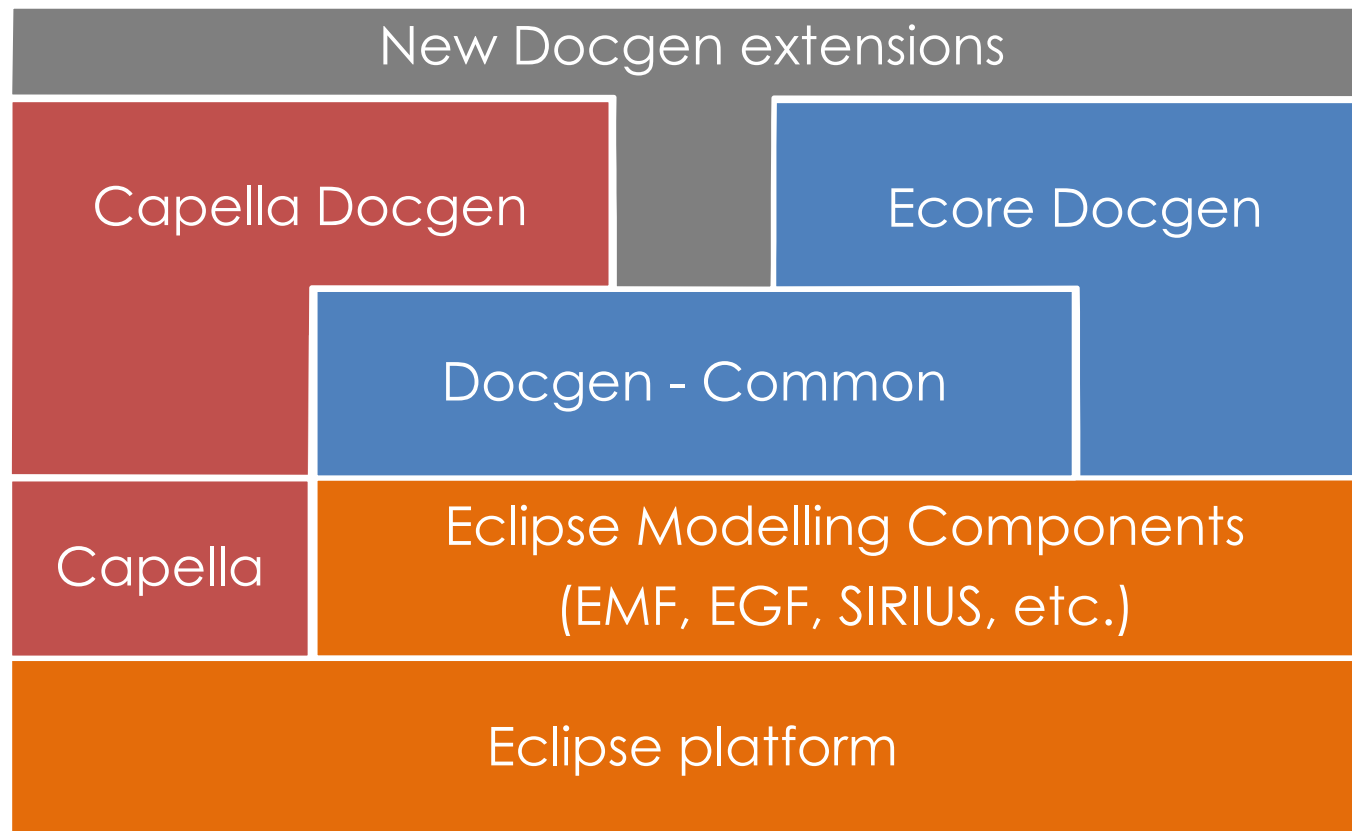
Operations

	Name	Return	Parameter(s)
	destroy	0..1	

Page generated: ven., 1 juil. 2016 14:54:29 CEST

Copyright © 2014 The Eclipse Foundation. All Rights Reserved.

Generated by Docgen.html from the [Kitalpha](#) Project



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

Capella Documentation Generation

In-Flight Entertainment System

[Search Index](#) | [Back to Index](#)

- In-Flight Entertainment System
 - Operational Analysis
 - System Analysis
 - Logical Architecture
 - Logical Functions
 - Requirements
 - Capabilities
 - Interfaces
 - Data
 - Logical Context
 - IFE System
 - IFE Operating Modes
 - Aircraft Front Servers : Aircraft Front Servers
 - Seat TV : Seat TV
 - Cabin Screen : Cabin Screen
 - Cabin Terminal : Cabin Terminal
 - Aircraft Front Servers
 - Seat TV
 - Cabin Screen
 - Cabin Terminal
 - Actors
 - Physical Architecture
 - EPBS Architecture

Cabin Screen

LogicalComponent

In-Flight Entertainment System > In-Flight Entertainment System > Logical Architecture > IFE System > Cabin Screen

No description.

Allocated Functions

- Play Video Stream on Cabin Screen

Incoming Component Exchanges

Exchange	Source	Description	Allocated Functional Exchanges	Allocated Exchange Items
Streaming Protocol	Streaming Server		<ul style="list-style-type: none"> Pause-Stop Notification Video Stream 	

Ports

- CP 1

Realizing Elements

- Cabin Video Unit SW

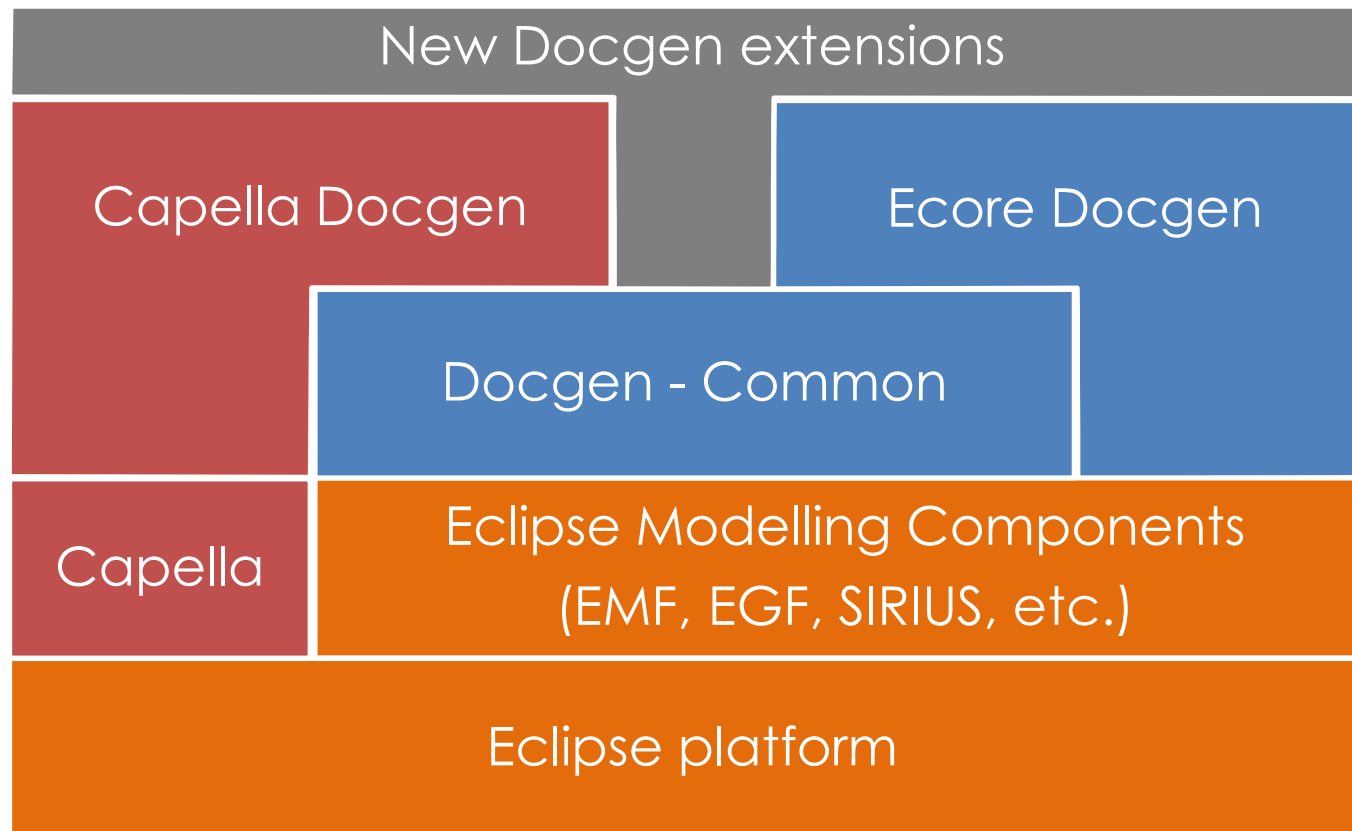
Diagrams displaying "Cabin Screen"

- II API IRI III DI All Components, Functions, CFs, FFs

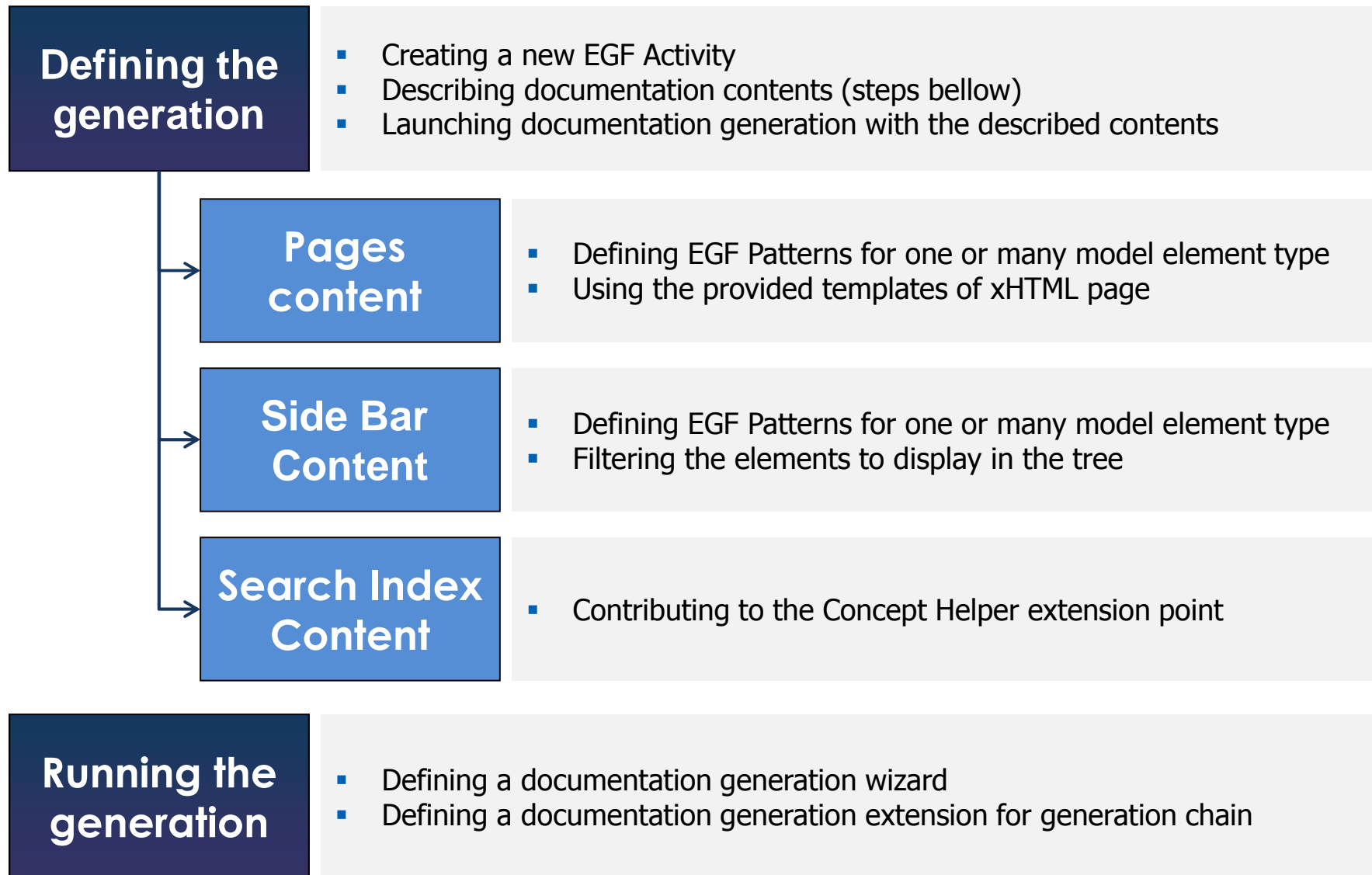
Page generated: ven., 1 juil. 2016 12:16:17 CEST

Copyright © 2014 The Eclipse Foundation. All Rights Reserved.

Generated by Doogen.html from the [Kitalpha Project](#)



This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.



Prerequisites:

1. *Eclipse RCP: you should be able to create a new Eclipse Plugin, an Eclipse Extension, etc.*
2. *EGF: you should be able to create a new EGF Activities, EGF Patterns, etc.*

Creating a new EGF Activity

1. Creating a new Eclipse plugin thanks to New project wizard
2. In that plugin, creating a new Folder with name = "egf"
3. In that folder, creating a new EGF Activity with root element = Factory Component
4. Defining the contracts (inputs values), EGF Patterns (HTML content) and the production plan

Documentation Content Definition:

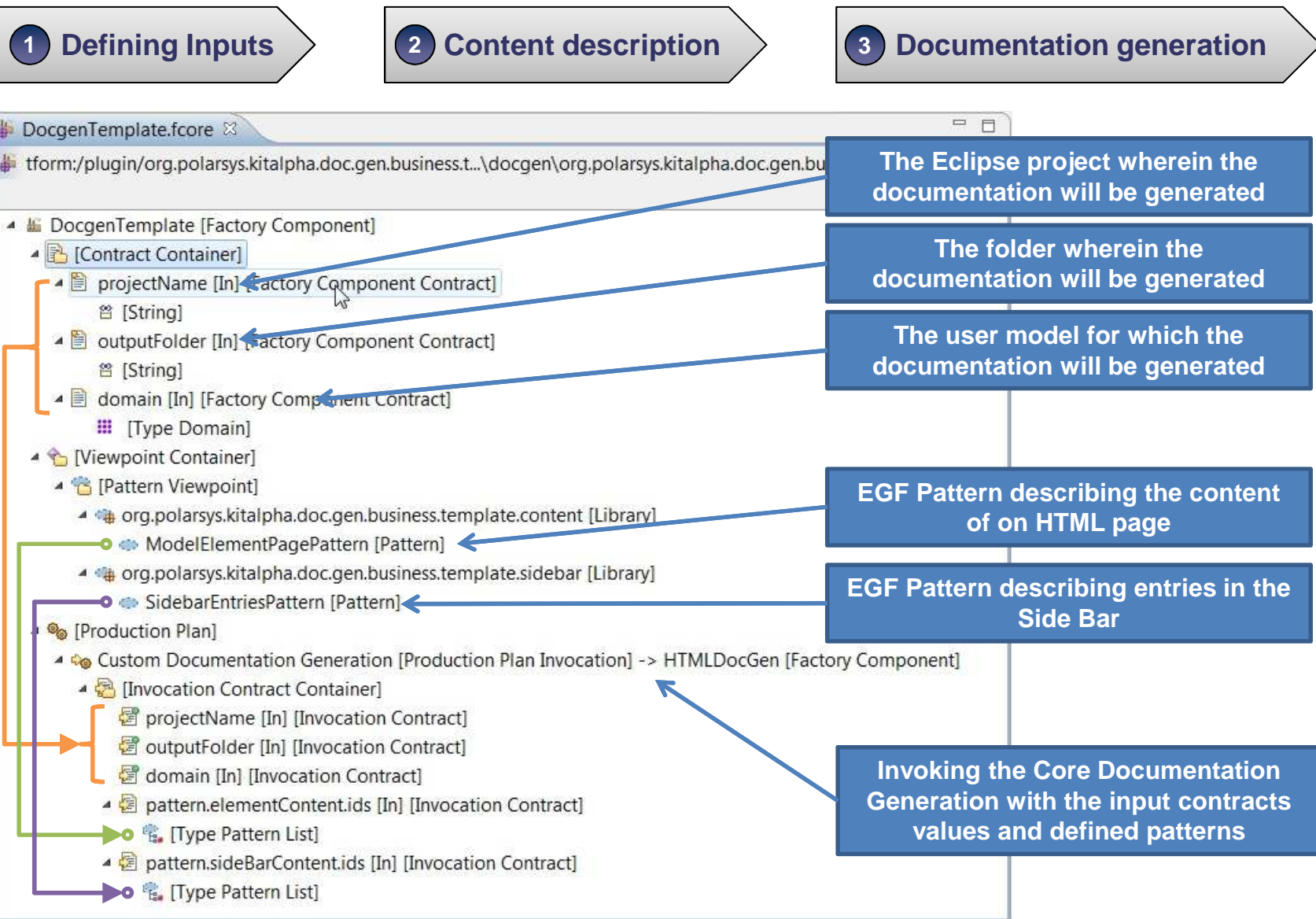
There are three kinds of content. All of them are generated by EGF Patterns.

1. Side Bar: contains the tree of your documentation
2. HTML pages: contains documentation of your model elements
3. Search Index: contains links to prepared HTML pages produced by the Search Engine.

Launching custom documentation generation

1. HTML pages generation is done by the Core documentation generation (the factory component HTMLDocGen)
2. Tooling should be provided to avoid launching EGF Activity manually

*The picture of the next page contains a template of a documentation generation Factory Component.
This example is available in the Kitalpha platform as an example.*



All EGF Patterns intended for HTML pages generation should inherit from ElementDocContent OR derived pattern

Each pattern should be of type JetNature

The screenshot shows the 'Specification' tab of the 'ModelElementPagePattern' editor. It features three main sections: 'Inheritance', 'Pattern Nature', and 'Parameters'.

Inheritance: A label 'Choose the super pattern:' is followed by a text field containing 'ElementDocContent'. Below it are 'Browse' and 'X' buttons. An arrow points from the 'All EGF Patterns...' annotation to this text field.

Pattern Nature: A label 'Select the kind of the pattern:' is followed by a dropdown menu showing 'JetNature'. An arrow points from the 'Each pattern should be of type JetNature' annotation to this dropdown.

Parameters: A label 'Define parameters for this pattern in the following section.' is above a table. The table has three columns: 'Name', 'Type', and 'Query'. The first row contains 'parameter' and 'EObject'. An arrow points from the 'Each pattern should be linked to one and only one EMF EClass' annotation to the 'EObject' cell. To the right of the table are five buttons: '+', '-', 'X', up arrow, and down arrow. An arrow points from the 'If a super pattern defined this value, derived patterns should not define it another time' annotation to the '-' button.

At the bottom, there are tabs for 'Overview', 'Specification', and 'Implementation'.

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

- Each pattern should redefines the three methods
- These methods should not be added in the orchestration section. They are already handled in the super pattern

The screenshot shows the 'Implementation' tab of a software development tool. It features several panels: 'Pattern methods' on the left, 'Implementation methods' in the center, 'Orchestration' on the right, and 'Variables' at the bottom left. Annotations with blue boxes and arrows point to specific elements:

- Pattern required Java classes import:** Points to the 'header' method in the 'Pattern methods' list.
- Pattern execution precondition:** Points to the 'init' method in the 'Pattern methods' list.
- Pattern methods:** Points to the 'setContext', 'setFileNameServ...', and 'content' methods in the 'Implementation methods' list.
- Pattern locals variables:** Points to the 'Variables' panel.
- Orchestration of pattern methods execution:** Points to the 'Organize method calls' section in the 'Orchestration' panel.

An orange box at the top contains a note: 'Each pattern should redefines the three methods. These methods should not be added in the orchestration section. They are already handled in the super pattern'. An arrow points from this note to the 'Implementation methods' list.

The 'Variables' panel includes a table for setting up variables:

Name	Type

The bottom of the interface has tabs for 'Overview', 'Specification', and 'Implementation', with 'Implementation' currently selected. An 'OPEN' button is located at the bottom center.

Methods implementation examples

setContext

```
<% fileName = fileNameService.getFileName(parameter); %>
```

Setting the file name of an HTML page under generation

setFileNameService

```
<% fileNameService = o.p.k.d.g.b.t.s.TemplateFileNameService.getInstance(); %>
```

Providing a File Name Service. That service defines the naming convention of HTML pages.

content

```
<%
/* Let generate a header for the page. */
/* First let get the Text provided by the ItemPrivoder */
String paramText = LabelProviderHelper.getText(parameter);

/* Second let get the Image provided by the ItemPrivoder.
 * To do that, we need projectName and outputFolder contract values.
 * We can get them as follow: */
String outputFolder = ctx.getValue("outputFolder").toString();
String projectName = ctx.getValue("projectName").toString();
String imagePath = LabelProviderHelper.getImageFileName(parameter,
                                                             projectName,
                                                             outputFolder);

%>
<h1>  <%=paramText%> </h1>
<h2> Static content </h2>
<p>
THIS IS A STATIC CONTENT <br/>
</p>
<% // Let generate now content built from the parameter element.
String objectToString = parameter.toString(); %>
<h2> Dynamic content </h2>
<%=objectToString%>
```

It is an implementation of the Java Interface IFileNameService. An example is shown in the next page.

Defining an HTML Content. That content will be placed in the Body of the HTML Page.

The data can be provided as static or computed from the parameter of the pattern or other sources.

IFilenameService implementation example

```
package org.polarsys.kitalpha.doc.gen.business.template.services;

import org.eclipse.emf.ecore.EObject;
import org.polarsys.kitalpha.doc.gen.business.core.util.DocGenHtmlUtil;
import org.polarsys.kitalpha.doc.gen.business.core.util.IFileNameService;
import org.polarsys.kitalpha.doc.gen.business.core.util.LabelProviderHelper;

/**
 * This class aims at defining the file naming convention to use when generating HTML pages
 */
public class TemplateFileNameService implements IFileNameService {

    private static final TemplateFileNameService INSTANCE = new TemplateFileNameService();

    public static TemplateFileNameService getInstance(){
        return INSTANCE;
    }

    @Override
    public String getFileName(EObject eObject) {
        String fileName = LabelProviderHelper.getText(eObject);
        return DocGenHtmlUtil.getValidFileName(fileName).toLowerCase();
    }
}
```

All EGF Patterns for side bar generation should inherit from ElementSideBarContent pattern

Pattern Nature should be JetNature

The screenshot shows the 'Specification' tab of a pattern editor. The 'Inheritance' section has 'Parent: ElementSideBarContent' selected. The 'Pattern Nature' section has 'Type: JetNature' selected. The 'Parameters' section contains a table with one row: 'parameter' of type 'EObject'. A blue arrow points from the 'EObject' cell to a callout box stating 'Pattern should be linked to one and only one EMF EClass'. The interface includes tabs for 'Overview', 'Specification', and 'Implementation' at the bottom.

Inheritance

Choose the super pattern:
Parent: ElementSideBarContent

Pattern Nature

Select the kind of the pattern:
Type: JetNature

Parameters

Define parameters for this pattern in the following section.

Name	Type	Query
parameter	EObject	

Buttons: +, -, ✖, ↕, ↕

Overview Specification Implementation

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

- Pattern should redefines the five methods
- These methods should not be added in the orchestration section. They are already handled in the super pattern

Pattern required
Java classes import

Pattern execution
precondition

DocgenTemplate.fcore SidebarEntriesPattern

Implementation

Pattern methods:

- header
- init
- preCondition
- footer

Implementation methods:

- setCurrentObject
- setFileNameServ...
- contentSidebarEl...
- startSidebarSubE...
- endSidebarSubEl...

Variables

Set up some variable available in all methods:

Name	Type

Orchestration

Organize method calls:

[Call to super pattern orchestration]

Overview Specification Implementation

Methods implementation examples

setCurrentObject

```
<% currentObject = parameter; %>
```

Setting the file name of an HTML page under generation

setFileNameService

```
<% fileNameService = o.p.k.d.g.b.t.s.TemplateFileNameService.getInstance(); %>
```

Providing a File Name Service. That service defines the naming convention of HTML pages. It is an implementation of the Java Interface IFileNameService.

contentSidebarElement

```
<%
String projectName = ctx.getValue("projectName").toString();
String outputFolder = ctx.getValue("outputFolder").toString();
String modelName = DocGenHtmlUtil.getModelName(currentObject);
String fileName = fileNameService.getFileName(currentObject);
String text = EscapeChars.forHTML(fileName);
String imageFileName = LabelProviderHelper.getImageFileName(currentObject,
                                                             projectName,
                                                             outputFolder);

%>
" />
<a href=" ../<%= modelName%>/<%=fileName%>.html" target="content">
  <%= text %>
</a>
```

Defining an HTML Content that will be used as an entry in the side bar.

That content is composed of an image and a link to a given generated HTML page.

startSidebarSubElement

Same implementation for endSidebarSubElement

```
<%
if(/*Replace condition*/ true) {
  super.method_startSidebarSubElement(stringBuffer, ctx);
}
%>
```

Checking if sub entries will be generated for a given side bar entry.

A condition should be defined for the IF bloc.

Based on the Extension Point “conceptshelper”

1. ID: *org.polarsys.kitalpha.doc.gen.business.core.conceptshelper*
2. Extensions should provide an implementation of the Java Interface `IConceptsHelper` (cf. example)

IConceptsHelper implementation example

```
package org.polarsys.kitalpha.doc.gen.business.template.ext;

import org.eclipse.emf.ecore.EObject;
import org.polarsys.kitalpha.doc.gen.business.core.helper.IConceptsHelper;
import org.polarsys.kitalpha.doc.gen.business.core.util.LabelProviderHelper;

public class TemplateConceptsHelper implements IConceptsHelper {
    public TemplateConceptsHelper() {
        // TODO Auto-generated constructor stub
    }

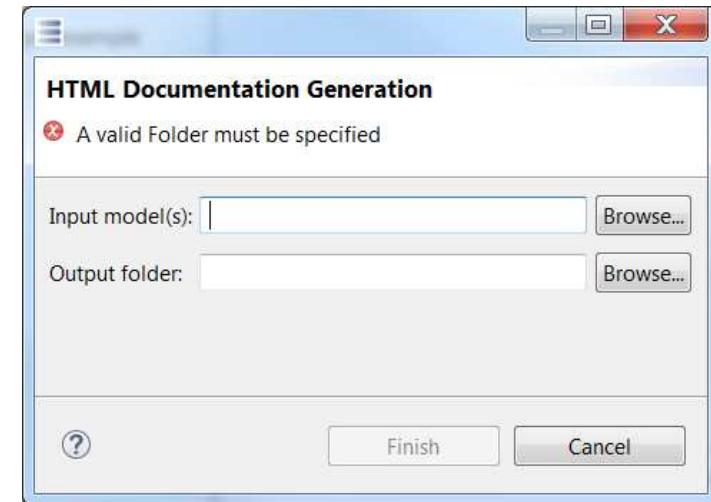
    @Override
    public boolean accept(Object modelElement) {
        /** A condition should be provided in order to select the model element to index. */
        return true; // This means that all model elements are indexed
    }

    @Override
    public String getConceptLabel(Object modelElement) {
        /** Return a label to display in the search index for the current model element */
        return LabelProviderHelper.getText((EObject) modelElement);
    }
}
```

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.

Documentation generation Wizard

1. An abstract wizard implementation is available in the Core component.
2. Extensions should provides
 - A custom implementation of that wizard
 - UI integration of that Custom Wizard
3. That wizard requires a documentation generation launcher (an EGF activity)



Generation chain extension :

See EGF Documentation http://wiki.eclipse.org/EGF_Tutorial_and_Use_Cases#Generation_Chain

1 Defining generation launcher

2 Defining generation Wizard

Generation launcher is an EGF Activity containing

1. Exactly two String input contracts: *projectName* and *outputFolder*
2. A Domain Viewpoint containing one and only one EMF Domain
3. A production plan containing only the invocation of the custom documentation generation Activity

The screenshot shows the Eclipse IDE interface. The left pane displays the project structure for `DocgenTemplateLauncher.fcore`. The tree view shows the following structure:

- `DocgenTemplateLauncher [Factory Component]`
 - `[Contract Container]`
 - `projectName [In] [Factory Component Contract]`
 - `[String]`
 - `outputFolder [In] [Factory Component Contract]`
 - `[String]`
 - `[Viewpoint Container]`
 - `[Domain Viewpoint]`
 - `[EMF Domain]`
 - `[Production Plan]`
 - `Docgen Template [Production Plan Invocation] -> DocgenTemplate [Factory Component]`
 - `[Invocation Contract Container]`
 - `projectName [In] [Invocation Contract]`
 - `outputFolder [In] [Invocation Contract]`
 - `domain [In] [Invocation Contract]`
 - `EMFDomain [Type Domain]`

The right pane shows the Properties window with the Overview tab selected. The table below represents the data shown in the Properties window:

Property	Value
Overview	
Description	
ID	_knAyYAiTEeaORIDodnmHg
Name	DocgenTemplateLauncher

1 Defining generation launcher

2 Defining generation Wizard

Abstract Wizard information

Plugin name	org.polarsys.kitalpha.doc.gen.business.core.ui
Java Package	org.polarsys.kitalpha.doc.gen.business.core.ui.wizards
Java class	HTMLDocumentationGenerationWizard.java
Description	<p>This class provides a wizard allowing end-users to run a given documentation generation. It asks end-user to select a model and to provide an output folder.</p> <p>It is an abstract class containing the abstract method <code>getLaunchersURI()</code>. Custom generation should provides an implementation for that method. It should return at least one launcher URI (The EGF activity defined in the previous step).</p> <p>An example is shown in the next page.</p>

HTMLDocumentationGenerationWizard implementation example

```
package org.polarsys.kitalpha.doc.gen.business.template.launcher.internal;

import java.util.HashMap;
import java.util.Map;

import org.polarsys.kitalpha.doc.gen.business.template.launcher.Activator;

import org.eclipse.emf.common.util.URI;
import org.polarsys.kitalpha.doc.gen.business.core.ui.wizards.HTMLDocumentationGenerationWizard;

public class TemplateHTMLDocumentationGenerationWizard extends HTMLDocumentationGenerationWizard {
    /** Launcher relative path */
    private static final String LAUNCHER_FILE_PATH = "/egf/DocgenTemplateLauncher.fcore";

    /** Main factory component ID */
    private static final String FACTORY_COMPONENT_ID = "_knAyYAiTEeaORIDodnrmHg";

    /** Launcher URI */
    private static final URI TEMPLATE_LAUNCHER_URI = URI.createURI("platform:/plugin/" +
        Activator.PLUGIN_ID + LAUNCHER_FILE_PATH + "#" + FACTORY_COMPONENT_ID);

    @Override
    protected Map<String, URI> getLaunchersURI() {
        Map<String, URI> result = new HashMap<String, URI>();
        result.put("Default Launcher", TEMPLATE_LAUNCHER_URI);
        return result;
    }
}
```

This document is not to be reproduced, modified, adapted, published, translated in any material form in whole or in part nor disclosed to any third party without the prior written permission of Thales. © THALES 2013 – All rights reserved.



Initiative supported by **Clarity**,
a French collaborative project
<http://www.clarity-se.org/>



<http://polarsys.org/kitalpha/>

<http://polarsys.org/capella/>